

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA: CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI TẬP
HỆ ĐIỀU HÀNH
BÀI THỰC HÀNH SỐ 3

Giảng viên hướng dẫn : Đinh Trường Duy
Nhóm môn học : 01
Tổ thực hành : 03
Sinh Viên : Đinh Thị Thanh Tâm – B22DCAT253

HÀ NỘI, THÁNG 11 NĂM 2024

MỤC LỤC

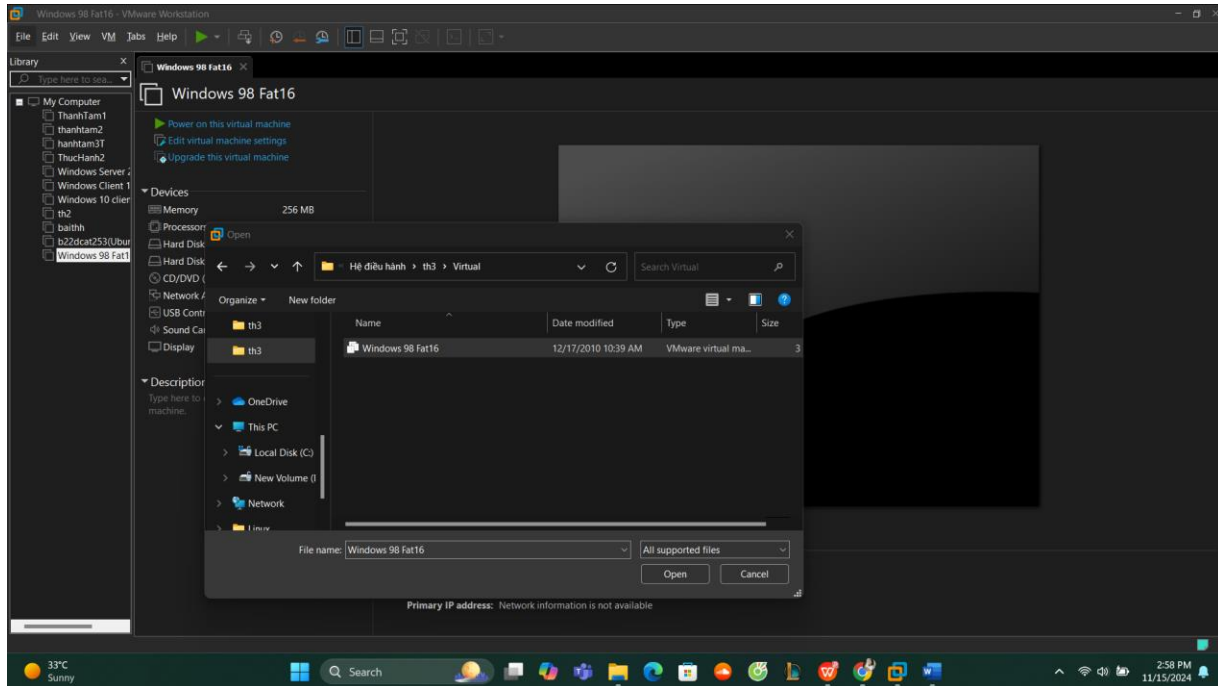
Chuẩn bị.....	3
Khởi động máy ảo:.....	3
1. Tùy chỉnh file FAT.cpp để đọc và thực hiện các yêu cầu	5
1.1. Viết đoạn chương trình in nội dung của 150 ô FAT đầu tiên của ổ đĩa D ra màn hình từ đó in ra bảng bit của 150 khối nhớ của ổ D.....	5
1.2. Viết chương trình đọc FAT vào bộ nhớ tại địa chỉ << int *fat >> . Giả sử một file được lưu trữ trên cluster đầu tiên là n. Viết chương trình in các cluster thuộc file đó và in ra tên của File đó trong ROOT.	6
1.3. Viết chương trình đọc thư mục gốc của hệ thống file FAT16 sử dụng tên file độ dài tối đa 8 ký tự được đọc vào bộ nhớ tại địa chỉ << void * root >>.....	7
2. Viết chương trình trên C/C++ để thực hiện các nội dung sau:.....	10
– Đọc và in thông tin từ BOOT.....	10
- Đọc, phân tích, hiển thị nội dung bảng FAT.....	11
- Đọc, phân tích, hiển thị ROOT.	14
- Duyệt số thứ tự hoặc nội dung các cluster của file cho trước	16
- Viết đoạn chương trình in ra nội dung giống như câu lệnh dir.	20
KẾT LUẬN	22

Chuẩn bị

Cần chuẩn bị cài đặt máy ảo, đĩa ảo, thẻ nhớ USB, lập trình các bài tập đọc FAT

Khởi động máy ảo:

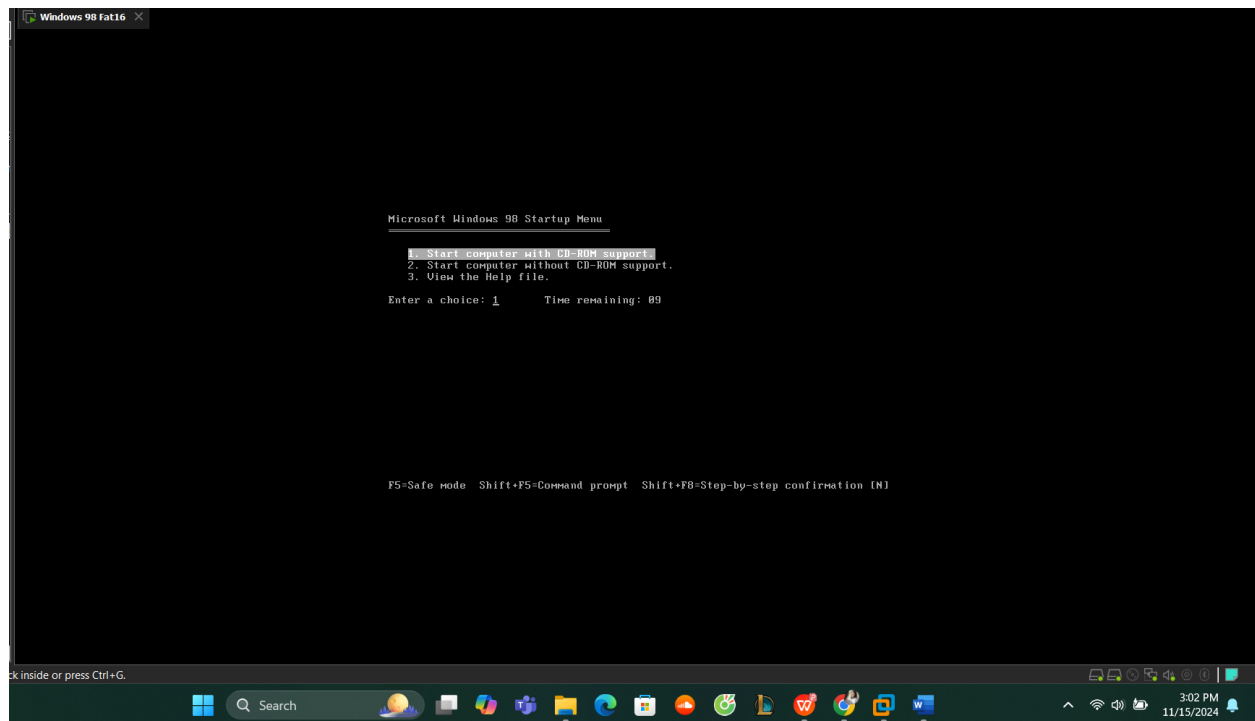
Trong giao diện của máy ảo VMware, chọn File => open mở file *Windows 98 Fat16* để thực hiện bài thực hành



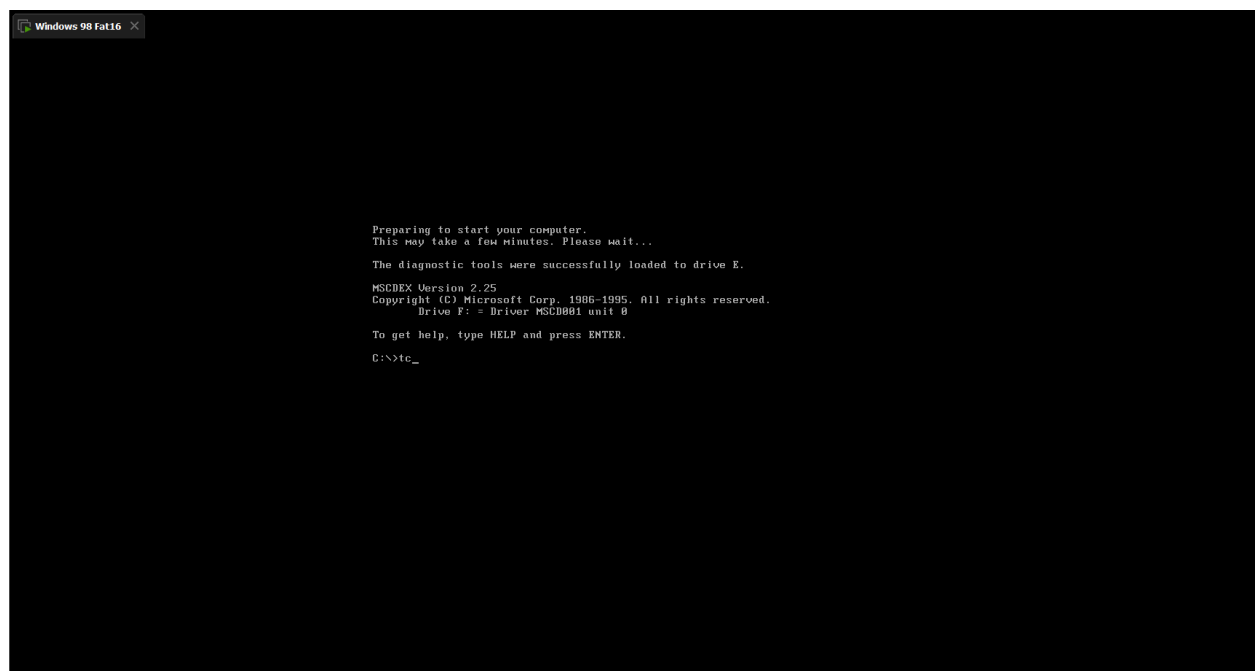
Windows 98 đang được khởi chạy:



Bấm Enter trên bàn phím để tiếp tục khởi chạy:

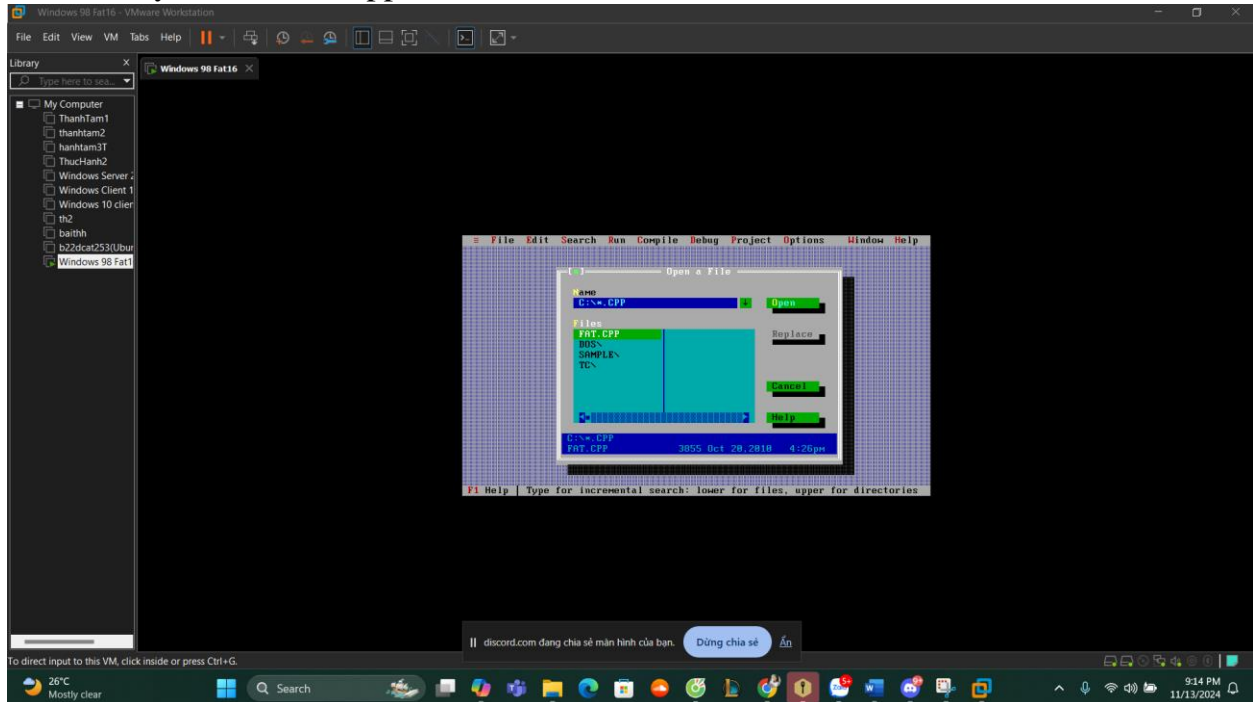


Vào thư mục C:\TC và khởi động Turbo C.



1. Tùy chỉnh file FAT.cpp để đọc và thực hiện các yêu cầu

Mở và chạy file C:\FAT.cpp



1.1. Viết đoạn chương trình in nội dung của 150 ô FAT đầu tiên của ổ đĩa D ra màn hình từ đó in ra bảng bit của 150 khối nhớ của ổ D.

- Viết đoạn chương trình in nội dung của 150 ô FAT đầu tiên của ổ đĩa D ra màn hình:

In 150 file FAT đầu tiên ta dùng 1 vòng lặp for cho chạy từ 0 đến 149 và in giá trị của mỗi ô FAT (fat[i])

```
printf("Content of first 15 FAT cells:");  
for (i = 0; i < 150; i++)  
    printf("%u ", fat[i]);
```

- Đoạn chương trình in ra bảng bit của 150 khối nhớ của ổ D:

Dùng 1 vòng lặp khác cho chạy từ 0 đến 149. Nếu giá trị của ô FAT tương ứng bằng 0 (fat[i] == 0) thì in ra màn hình giá trị là 1, ngược lại thì in ra 0.

```

//printing first 150 FAT cells
printf("Content of first 15 FAT cells:");
for (i = 0; i < 150; i++)
    printf("%u ", fat[i]);

printf("\n");
printf("in bang bit 150 fat shell \n");
for(int i = 0; i<150; i++){
    if(fat[i] == 1) printf("0 ");
    else printf("1 ");
}

```

Kết quả

```

-----
FAT size: 33
Reserved: 6
Content of first 15 FAT cells:65528 65535 3 4 5 6 7 8 9 65535 11 12 13 14 15 16
17 18 65535 20 21 22 23 24 25 26 65535 28 29 65535 31 65535 33 34 35 36 37 38 65
535 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
in bang bit 150 fat shell
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Number of free clusters from first 100 clusters:61
Clusters of a file from 5: ->5->6->7->8->9

Reading ROOT information:
-----
3 first items of root:
README  2      15608
FILELIST 10     18019
README  19     15481

Clusters belong to file readme:
Enter a file name:README_

```

1.2. Viết chương trình đọc FAT vào bộ nhớ tại địa chỉ << int *fat >> . Giả sử một file được lưu trữ trên cluster đầu tiên là n. Viết chương trình in các cluster thuộc file đó và in ra tên của File đó trong ROOT.

Nhập số nguyên n là cluster đầu tiên của file. Nếu cluster đầu tiên của file bằng n thì in ra tên của file

```

File Edit Search Run Compile Debug Project Options Window Help
FAT.CPP 1=[↕]

//Printing clusters of a file from cluster n
unsigned int n ;
int file = 1;
//unsigned int cur = n;
printf("Cluster first of file: ");
scanf("%d", &n);
unsigned int cur = n;

printf("Clusters of a file from %u: ",n);
if(fat[cur] == 0 ) {
    printf("file not found");
    file =0;
}
else {
    while(cur < 0xFFFF){
        printf("->%u", cur);
        cur = fat[cur];
    }
}

* 107:10
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```

Kết quả

```

Cluster first of file: 10
Clusters of a file from 10: ->10->11->12->13->14->15->16->17->18

Reading ROOT information:
-----
3 first items of root:
README  2      15608
FILELIST 10     18019
README  19     15481

Clusters belong to file readme:
Enter a file name:

```

1.3. Viết chương trình đọc thư mục gốc của hệ thống file FAT16 sử dụng tên file độ dài tối đa 8 ký tự được đọc vào bộ nhớ tại địa chỉ << void *root >>.

Viết chương trình C/C++ thực hiện hai việc:

- in tên và độ dài các file trong thư mục gốc;
- tìm một file có tên cho trước trong thư mục gốc và cho biết file đó có bao nhiêu khối nhớ.

Chỉnh sửa đoạn mã

```
//Printing first 3 items of root
printf("3 first items of root:\n");
for(i = 0; i < 3; i++){
    if(root[i].name[0] == ' ') continue;
    for(int j = 0; j < 8 && root[i].name[j] != ' '; j++){
        printf("%c", root[i].name[j]);
    }
    printf("\t");
    printf("%d \t %d\n", root[i].first_cluster, root[i].size);
}

File Edit Search Run Compile Debug Project Options Window Help
FAT.CPP 1=[↓]

//Printing all items of root
printf("All items of root:\n");
for(i = 0; i < boot.ROOT_size; i++){
    if(root[i].size != 0){
        for(int j = 0; j < 8 ; j++){
            printf("%c", root[i].name[j]);
        }

        printf("\t, %d\n", root[i].size);
    }
}

//Printing clusters belonging to the file
int count_cluster = 0;
if(first_cluster >= 0){
    cur = first_cluster;
    while(cur < 0xFFF8){
        // printf("%u ", cur);
        count_cluster++;
        cur = fat[cur];
    }
}
printf("So khoi cua file la: %d\n", count_cluster);
free(root);
free(fat);

getchar();
}

178:17
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Kết quả

[illegible]

Clusters of a file from 5: ->5->6->7->8->9

```

README          , 15608
FILELIST        , 18019
README          , 15481
README          , 4217
FAT              , 3743
FAT              , 12803

```

```
Enter a file name:README
```

2. Viết chương trình trên C/C++ để thực hiện các nội dung sau:

- *Đọc và in thông tin từ BOOT.*

Khai báo kiểu dữ liệu cho các biến trong file BOOT.cpp

```
File Edit Search Run Compile Debug Project Options Window Help
BOOT.CPP
struct BOOT {
    char jmp[3];
    char OEM[8];
    int bytes_per_sector;
    char sectors_per_cluster;
    int reserved;
    char FAT_cnt;
    int ROOT_size;
    int total_sectors;
    char media;
    int FAT_size;
    int sectors_per_track;
    int head_cnt;
    long hidden_sectors;
    long total_sectors_long;
    char unknown[3];
    long serial;
    char volume[11];
    char FAT_type[8];
    char loader[448];
    char mark[2];
}
27:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Đọc dữ liệu từ file BOOT

```
File Edit Search Run Compile Debug Project Options Window Help
BOOT.CPP
printf("Thông tin Boot Sector:\n");
printf("-----\n");
printf("Mã nhảy: %02X %02X %02X\n", boot.jmp[0], boot.jmp[1], boot.jmp[2]);
printf("Nhãn OEM: %.8s\n", boot.OEM);
printf("Số byte trên mỗi sector: %d\n", boot.bytes_per_sector);
printf("Số sector trên mỗi cluster: %d\n", boot.sectors_per_cluster);
printf("Số sector dự trữ: %d\n", boot.reserved);
printf("Số bảng FAT: %d\n", boot.FAT_cnt);
printf("Số mục trong thư mục ROOT: %d\n", boot.ROOT_size);
printf("Tổng số sector (nếu nhỏ hơn 65535): %d\n", boot.total_sectors);
printf("Định danh phương tiện lưu trữ: %02X\n", boot.media);
printf("Kích thước mỗi bảng FAT: %d\n", boot.FAT_size);
printf("Số sector trên mỗi track: %d\n", boot.sectors_per_track);
printf("Số đầu đọc trên ổ đĩa: %d\n", boot.head_cnt);
printf("Số sector ẩn: %ld\n", boot.hidden_sectors);
printf("Tổng số sector (nếu lớn hơn 65535): %ld\n", boot.total_sectors_long);
printf("Phần không xác định: %02X %02X %02X\n", boot.unknown[0], boot.unknown[1], boot.unknown[2]);
printf("Số serial của đĩa: %08lX\n", boot.serial);
printf("Nhãn của volume: %.11s\n", boot.volume);
printf("Loại hệ thống FAT: %.8s\n", boot.FAT_type);
printf("Dấu kết thúc sector khởi động: %02X %02X\n", boot.mark[0], boot.mark[1]);
62:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Kết quả thông tin BOOT:

```
C:\>tc
Thông tin Boot Sector:
-----
Mã nhảy: FFEB 3C FF90
Nhãn OEM: MSWIN4.0
Số byte trên mỗi sector: 512
Số sector trên mỗi cluster: 4
Số sector dự trữ: 6
Số bảng FAT: 2
Số mục trong thư mục ROOT: 512
Tổng số sector (nếu nhỏ hơn 65535): -32587
Định danh phương tiện lưu trữ: 0xFF8
Kích thước mỗi bảng FAT: 33
Số sector trên mỗi track: 63
Số đầu đọc trên ổ đĩa: 4
Số sector ẩn: 63
Tổng số sector (nếu lớn hơn 65535): 32949
Phân không xác định: FF80 00 29
Số serial của đĩa: 4CB0D9F3
Nhãn của volume:
Loại hệ thống FAT: FAT16
Đầu kết thúc sector khởi động: 55 FFAA
-----

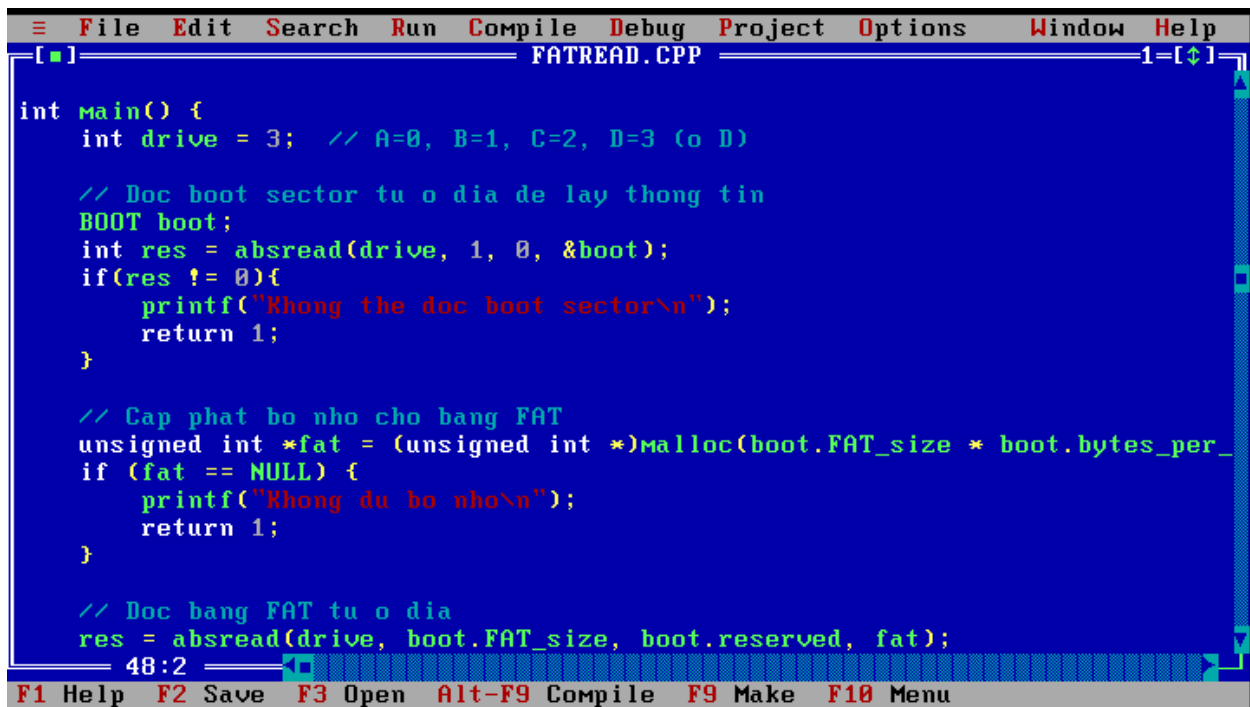
C:\>_
```

- *Đọc, phân tích, hiển thị nội dung bảng FAT.*

Khai báo thông tin các dữ liệu trong file FAT16

```
File Edit Search Run Compile Debug Project Options Window Help
[ ] FATREAD.CPP 1=[ ]
struct BOOT { // Cấu trúc boot cho FAT16
    char jmp[3];
    char OEM[8];
    int bytes_per_sector;
    char sectors_per_cluster;
    int reserved;
    char FAT_cnt;
    int ROOT_size;
    int total_sectors;
    char media;
    int FAT_size;
    int sectors_per_track;
    int head_cnt;
    long hidden_sectors;
    long total_sectors_long;
    char unknown[3];
    long serial;
    char volume[11];
    char FAT_type[8];
    char loader[448];
    char mark[2];
}
26:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Đọc thông tin cần thiết từ boot sector, cấp phát bộ nhớ



```
File Edit Search Run Compile Debug Project Options Window Help
FATREAD.CPP 1=[↕]

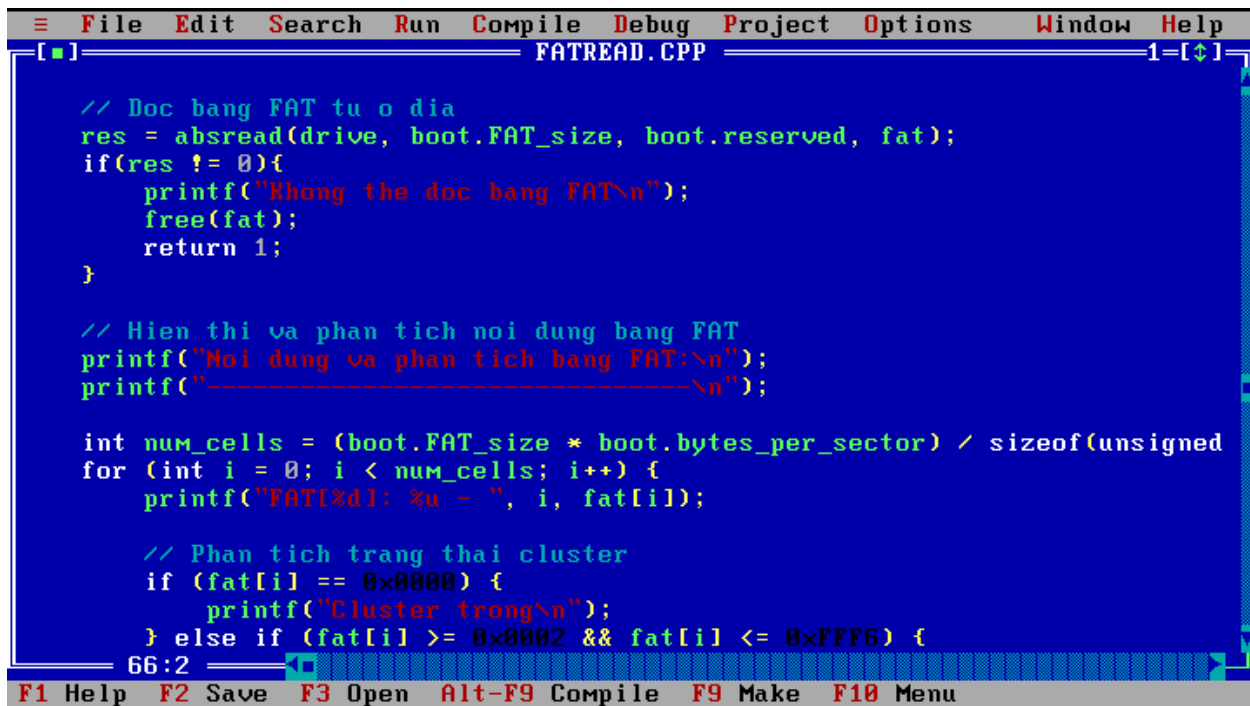
int main() {
    int drive = 3; // A=0, B=1, C=2, D=3 (o D)

    // Doc boot sector tu o dia de lay thong tin
    BOOT boot;
    int res = absread(drive, 1, 0, &boot);
    if(res != 0){
        printf("Khong the doc boot sector\n");
        return 1;
    }

    // Cap phat bo nho cho bang FAT
    unsigned int *fat = (unsigned int *)malloc(boot.FAT_size * boot.bytes_per_
    if (fat == NULL) {
        printf("Khong du bo nho\n");
        return 1;
    }

    // Doc bang FAT tu o dia
    res = absread(drive, boot.FAT_size, boot.reserved, fat);
    48:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Đọc bảng FAT , hiển thị và phân tích nội dung từ ổ đĩa



```
File Edit Search Run Compile Debug Project Options Window Help
FATREAD.CPP 1=[↕]

    // Doc bang FAT tu o dia
    res = absread(drive, boot.FAT_size, boot.reserved, fat);
    if(res != 0){
        printf("Khong the doc bang FAT\n");
        free(fat);
        return 1;
    }

    // Hien thi va phan tich noi dung bang FAT
    printf("Noi dung va phan tich bang FAT:\n");
    printf("-----\n");

    int num_cells = (boot.FAT_size * boot.bytes_per_sector) / sizeof(unsigned
    for (int i = 0; i < num_cells; i++) {
        printf("FAT[%d]: %u - ", i, fat[i]);

        // Phan tich trang thai cluster
        if (fat[i] == 0x0000) {
            printf("Cluster trong\n");
        } else if (fat[i] >= 0x0002 && fat[i] <= 0xFFF6) {
            66:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

```
File Edit Search Run Compile Debug Project Options Window Help
FATREAD.CPP
// Phan tich trang thai cluster
if (fat[i] == 0x0000) {
    printf("Cluster trong\n");
} else if (fat[i] >= 0x0002 && fat[i] <= 0xFFF6) {
    printf("Cluster da cap phat, chi den cluster tiep theo: 2u\n", fat[i]);
} else if (fat[i] >= 0xFFF8 && fat[i] <= 0xFFFF) {
    printf("Cluster cuoi cung cua tep\n");
} else {
    printf("Trang thai khong xac dinh\n");
}

if (i % 16 == 15) { // Sau moi 16 cell, xuong dong
    printf("\n");
}

// Giai phong bo nho
free(fat);

return 0;
}

83:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu
```

Kết quả

```
Noi dung va phan tich bang FAT:
-----
FAT[0]: 65528 - Cluster cuoi cung cua tep
FAT[1]: 65535 - Cluster cuoi cung cua tep
FAT[2]: 3 - Cluster da cap phat, chi den cluster tiep theo: 3
FAT[3]: 4 - Cluster da cap phat, chi den cluster tiep theo: 4
FAT[4]: 5 - Cluster da cap phat, chi den cluster tiep theo: 5
FAT[5]: 6 - Cluster da cap phat, chi den cluster tiep theo: 6
FAT[6]: 7 - Cluster da cap phat, chi den cluster tiep theo: 7
FAT[7]: 8 - Cluster da cap phat, chi den cluster tiep theo: 8
FAT[8]: 9 - Cluster da cap phat, chi den cluster tiep theo: 9
FAT[9]: 65535 - Cluster cuoi cung cua tep
FAT[10]: 11 - Cluster da cap phat, chi den cluster tiep theo: 11
FAT[11]: 12 - Cluster da cap phat, chi den cluster tiep theo: 12
FAT[12]: 13 - Cluster da cap phat, chi den cluster tiep theo: 13
FAT[13]: 14 - Cluster da cap phat, chi den cluster tiep theo: 14
FAT[14]: 15 - Cluster da cap phat, chi den cluster tiep theo: 15
FAT[15]: 16 - Cluster da cap phat, chi den cluster tiep theo: 16

FAT[16]: 17 - Cluster da cap phat, chi den cluster tiep theo: 17
FAT[17]: 18 - Cluster da cap phat, chi den cluster tiep theo: 18
FAT[18]: 65535 - Cluster cuoi cung cua tep
FAT[19]: 20 - Cluster da cap phat, chi den cluster tiep theo: 20

C:\>
```

- *Đọc, phân tích, hiển thị ROOT.*

Khai báo thông tin BOOT

```

File Edit Search Run Compile Debug Project Options Window Help
FATREAD.CPP 1
ROOT.CPP 2=[↑]
[■]
struct BOOT {
    char jmp[31];
    char OEM[81];
    int bytes_per_sector;
    char sectors_per_cluster;
    int reserved;
    char FAT_cnt;
    int ROOT_size;
    int total_sectors;
    char media;
    int FAT_size;
    int sectors_per_track;
    int head_cnt;
    long hidden_sectors;
    long total_sectors_long;
    char unknown[31];
    long serial;
    char volume[111];
    char FAT_type[81];
    char loader[4481];
}
26:1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```

Khai báo thông tin ROOT

```

File Edit Search Run Compile Debug Project Options Window Help
FATREAD.CPP 1
ROOT.CPP 2=[↑]
[■]
struct ROOT {
    char name[81];
    char ext[31];
    char attr;
    char reserved[10];
    char time[21];
    char date[21];
    int first_cluster;
    long size;
};

```

Đọc boot sector từ ổ đĩa

```
File Edit Search Run Compile Debug Project Options Window Help
ROOT.CPP
void main() {
    int drive = 3; // Dung o dia D (A=0, B=1, C=2, D=3 ...)
    BOOT boot;

    // Doc boot sector tu o dia
    int res = absread(drive, 1, 0, &boot);
    if(res != 0) {
        printf("Khong the doc boot sector\n");
        return;
    }
}
```

Tính toán kích thước thư mục gốc ROOT và vị trí bắt đầu của nó

```
int root_size_in_bytes = boot.ROOT_size * 32; // Moi muc thu muc goc chiem
int root_start_sector = boot.reserved + boot.FAT_cnt * boot.FAT_size;
```

Sau khi tính toán, cấp phát bộ nhớ cho thư mục root

```
ROOT *root = (ROOT *)malloc(root_size_in_bytes);
if(root == NULL) {
    printf("Khong du bo nho\n");
    return;
}
61:1
```

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Đọc thư mục root từ đĩa, nếu không thể đọc thì thoát chương trình

```
File Edit Search Run Compile Debug Project Options Window Help
ROOT.CPP
// Doc thu muc goc tu dia
int root_sector_count = root_size_in_bytes / boot.bytes_per_sector;
res = absread(drive, root_sector_count, root_start_sector, root);
if(res != 0) {
    printf("Khong the doc ROOT\n");
    free(root);
    return;
}
```

Hiển thị 10 thư mục đầu tiên trong thư mục gốc (root) với các thông tin: tên, kích thước, cluster đầu tiên

```

// Hien thi 10 muc dau tien trong ROOT:
printf("10 muc dau tien trong ROOT:\n");
printf("Ten Tep\tMo rong\tKich thuoc (byte)\tCluster dau tien\n");
printf("-----\n");

for(int i = 0; i < 10; i++) {
    if(root[i].name[0] == 0x00) break; // Neu gap muc rong, dung lai
    if(root[i].name[0] == 0xE5) continue; // Bo qua muc da xoa

    // Hien thi ten tep
    for(int j = 0; j < 8 && root[i].name[j] != ' '; j++)
        printf("%c", root[i].name[j]);
    printf("\t");

    // Hien thi mo rong tep
    for(int j = 0; j < 3; j++)
        printf("%c", root[i].ext[j]);
    printf("\t");

    // Hien thi kich thuoc tep va cluster dau tien
    printf("%ld\t%ld\n", root[i].size, root[i].first_cluster);
}

// Giai phong bo nho
free(root);

getch(); // Dung man hinh cho nguoi dung nhan phim
}

```

100:1

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

Kết quả:

```

10 muc dau tien trong ROOT:
Ten Tep Mo rong Kich thuoc (byte)      Cluster dau tien
-----
README  TXT      15608      2
FILELIST DOC      18019     10
README      15481     19
README  COM      4217      27
FAT      OBJ      3743      30
FAT      EXE      12803     32
-

```

- Duyệt số thứ tự hoặc nội dung các cluster của file cho trước

Cấu trúc thông tin của tệp trong boot


```
File Edit Search Run Compile Debug Project Options Window Help
[.] CLUSTER.CPP 1=[+]
struct BOOT { // Cau truc boot cho FAT16
    char jmp[3];
    char OEM[8];
    int bytes_per_sector;
    char sectors_per_cluster;
    int reserved;
    char FAT_cnt;
    int ROOT_size;
    int total_sectors;
    char media;
    int FAT_size;
    int sectors_per_track;
    int head_cnt;
    long hidden_sectors;
    long total_sectors_long;
    char unknown[3];
    long serial;
    char volume[11];
    char FAT_type[8];
    char loader[448];
    char mark[2];
}
27:1
```

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Cấu trúc thông tin của tệp trong root

```
File Edit Search Run Compile Debug Project Options Window Help
[.] CLUSTER.CPP 1=[+]
struct ROOT { // Cau truc thong tin cua tep trong ROOT
    char name[8];
    char ext[3];
    char attr;
    char reserved[10];
    char time[2];
    char date[2];
    int first_cluster;
    long size;
};
```

Đọc boot sector từ ổ đĩa

```
int main() {
    int drive = 3; // A=0, B=1, C=2, D=3 (o D)

    // Doc boot sector tu o dia
    BOOT boot;
    int res = absread(drive, 1, 0, &boot);
    if(res != 0){
        printf("Khong the doc boot sector\n");
        return 1;
    }
}
50:1
```

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

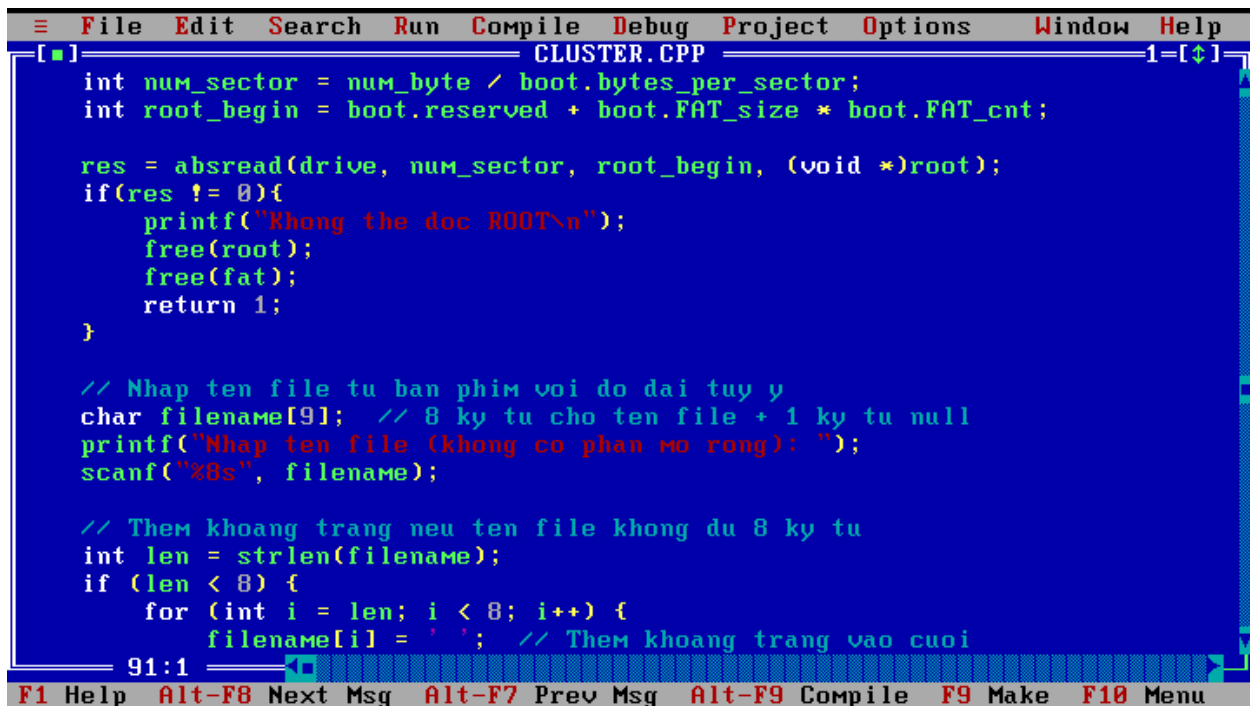
Đọc bảng FAT

```
unsigned int *fat = (unsigned int *)malloc (boot.FAT_size * boot.bytes_per  
if (fat == NULL) {  
    printf("Khong du bo nho\n");  
    return 1;  
}  
  
res = absread(drive, boot.FAT_size, boot.reserved, fat);  
if(res != 0){  
    printf("Khong the doc FAT\n");  
    free(fat);  
    return 1;  
}
```

Đọc root

```
int num_byte = boot.ROOT_size * 32; // kích thước của ROOT (32 byte cho m  
ROOT *root = (ROOT *)malloc(num_byte);  
if(root == NULL) return 1;  
  
int num_sector = num_byte / boot.bytes_per_sector;  
int root_begin = boot.reserved + boot.FAT_size * boot.FAT_cnt;  
  
res = absread(drive, num_sector, root_begin, (void *)root);  
if(res != 0){  
    printf("Khong the doc ROOT\n");  
    free(root);  
    free(fat);  
    return 1;  
}
```

Nhập tên file từ bàn phím với độ dài tùy ý



```
File Edit Search Run Compile Debug Project Options Window Help  
[■] CLUSTER.CPP 1=[  
int num_sector = num_byte / boot.bytes_per_sector;  
int root_begin = boot.reserved + boot.FAT_size * boot.FAT_cnt;  
  
res = absread(drive, num_sector, root_begin, (void *)root);  
if(res != 0){  
    printf("Khong the doc ROOT\n");  
    free(root);  
    free(fat);  
    return 1;  
}  
  
// Nhập tên file từ bàn phím với độ dài tùy ý  
char filename[91]; // 8 ký tự cho tên file + 1 ký tự null  
printf("Nhập tên file (không có phân họ riêng): ");  
scanf("%8s", filename);  
  
// Thêm khoảng trắng nếu tên file không đủ 8 ký tự  
int len = strlen(filename);  
if (len < 8) {  
    for (int i = len; i < 8; i++) {  
        filename[i] = ' '; // Thêm khoảng trắng vào cuối  
    }  
}  
91:1  
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu
```

Thêm khoảng trắng nếu tên file không đủ 8 ký tự

```
File Edit Search Run Compile Debug Project Options Window Help
CLUSTER.CPP
// Them khoang trang neu ten file khong du 8 ky tu
int len = strlen(filename);
if (len < 8) {
    for (int i = len; i < 8; i++) {
        filename[i] = ' '; // Them khoang trang vao cuoi
    }
    filename[8] = '\0'; // Ket thuc chuoi
}
```

Tìm file trong vùng root

```
int i, k;
char str[9];
int first_cluster = -1;
for(i = 0; i < boot.ROOT_size; i++){
    // Sao chép root[i].name sang str và thêm ký tự null để tạo chuỗi
    for(k = 0; k < 8 && root[i].name[k] != ' '; k++)
        str[k] = root[i].name[k];
    str[k] = 0;

    // So sánh tên file
    if(strncmp(root[i].name, filename, 8) == 0){
        107:1
    }
}
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu
```

So sánh tên file để biết file có tồn tại trong root không

```
if(strncmp(root[i].name, filename, 8) == 0){
    first_cluster = root[i].first_cluster;
    break;
}
}
```

In ra các cluster của file nếu tìm thấy

```
if(first_cluster >= 0){
    printf("Các cluster thuộc về file %s: ", filename);
    unsigned int cur = first_cluster;
    while(cur < 0xFFF8){
        printf("%u ", cur);
        cur = fat[cur];
    }
    printf("\n");
} else {
    printf("Không tìm thấy file %s\n", filename);
}
```

Giải phóng bộ nhớ

```

    free(root);
    free(fat);

    return 0;
}

```

111:1

F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Kết quả

```

C:\>fc
10 mục đầu tiên trong ROOT:
Ten Tep Mo rong Kich thuoc (byte) Cluster dau tien
-----
README TXT 15608 2
FILELIST DOC 18019 10
README 15481 19
README COM 4217 27
FAT OBJ 3743 30
FAT EXE 12803 32
Nhap ten file (khong co phan mo rong): FAT
Cac cluster thuoc ve file FAT : 30 31

```

- *Viết đoạn chương trình in ra nội dung giống như câu lệnh dir.*

Hàm hiển thị thông tin : tên, kích thước, ngày giờ của file

```

File Edit Search Run Compile Debug Project Options Window Help
[ ] DIR.CPP 2=[ ]
void displayFileInfo(struct ffblk file) {
    cout << file.ff_name << "\t";

    if (file.ff_attrib & FA_DIR) {
        cout << "<DIR>\t\t";
    } else {
        cout << file.ff_fsize << " bytes\t";
    }

    int year = ((file.ff_fdate >> 9) & 0x7F) + 1980;
    int month = (file.ff_fdate >> 5) & 0x0F;
    int day = file.ff_fdate & 0x1F;
    int hour = (file.ff_ftime >> 11) & 0x1F;
    int minute = (file.ff_ftime >> 5) & 0x3F;

    char period = (hour >= 12) ? 'p' : 'a';
    if (hour > 12) hour -= 12;
    if (hour == 0) hour = 12;

    cout << (month < 10 ? "0" : "") << month << "-"
        << (day < 10 ? "0" : "") << day << "-"
        << year << " "
        << (hour < 10 ? "0" : "") << hour << ":"
        << (minute < 10 ? "0" : "") << minute << period << "H" << endl;
}

```

Hàm main()

```
int main() {
    struct fblk file;
    int done;
    long total_size = 0;
    int file_count = 0;
    int dir_count = 0;

    cout << " Volume in drive C has no label." << endl;
    cout << " Directory of C:\\*.*)" << endl << endl;

    done = findfirst("*.*)", &file, FA_ARCH | FA_DIREC);

    while (!done) {
        displayFileInfo(file);
        if (file.ff_attrib & FA_DIREC) {
            dir_count++;
        } else {
            total_size += file.ff_fsize;
            file_count++;
        }
        done = findnext(&file);
    }

    cout << "          " << file_count << " file(s)\t" << total_size <
    cout << "          " << dir_count << " dir(s)\t" << "95,219,712 bytes"

    getch();
    return 0;
}
```

Kết quả

```
TC0000.SWP      262144 bytes    10-28-2010    2:04pm
FAT.OBJ 4118 bytes    10-28-2010    2:04pm
FAT.EXE 15295 bytes   10-28-2010    2:04pm
TC0001.SWP      262144 bytes    10-28-2010    3:10pm
DIR.CPP 1856 bytes    11-04-2024    5:03am
BOOT.BAK        3251 bytes    11-02-2024   12:04pm
CLUSTER.CPP     3396 bytes    11-13-2024    3:09pm
FATREAD.CPP     2214 bytes    11-13-2024    3:14pm
ROOT.BAK        2683 bytes    11-02-2024   12:04pm
TC0002.SWP      262144 bytes    11-15-2024    2:43pm
BOOT.OBJ        2588 bytes    11-15-2024    2:25pm
BOOT.EXE       10896 bytes    11-15-2024    2:25pm
BOOT.CPP        2152 bytes    11-15-2024    2:25pm
FATREAD.OBJ     2158 bytes    11-15-2024    2:30pm
FATREAD.EXE    10550 bytes    11-15-2024    2:30pm
ROOT.OBJ        2454 bytes    11-15-2024    2:37pm
ROOT.EXE       13565 bytes    11-15-2024    2:37pm
ROOT.CPP        2657 bytes    11-15-2024    2:37pm
CLUSTER.OBJ     2909 bytes    11-15-2024    2:42pm
CLUSTER.EXE    14350 bytes    11-15-2024    2:42pm
DIR.OBJ 15326 bytes    11-15-2024    2:46pm
DIR.EXE 30783 bytes    11-15-2024    2:46pm
          46 file(s)          1850906 bytes
          3 dir(s) 95,219,712 bytes free
```

KẾT LUẬN

Các nhiệm vụ chính của bài thực hành số 3 bao gồm: đọc và hiển thị thông tin hệ thống file FAT16, phân tích bảng FAT, quản lý thư mục gốc (ROOT), và triển khai các chương trình thực nghiệm trên C/C++ để xử lý dữ liệu từ hệ thống file. Các kết quả đạt được đã chứng minh được khả năng ứng dụng lý thuyết vào thực tiễn, từ việc phân tích cấu trúc dữ liệu FAT đến viết mã để tương tác với hệ thống file.

Quá trình thực hiện bài tập đã cung cấp thêm hiểu biết sâu sắc về cơ chế hoạt động của hệ thống file FAT16, từ boot sector, bảng FAT, đến thư mục gốc. Các bài tập lập trình đã giúp củng cố kiến thức và phát triển kỹ năng lập trình, đồng thời rèn luyện tư duy giải quyết vấn đề thực tế trong lĩnh vực bảo mật thông tin.

Mặc dù đã đạt được những kết quả quan trọng, nhưng nhận thấy một số hạn chế trong việc tối ưu hóa mã nguồn và xử lý các tình huống đặc biệt.